

STAT 5443, FINAL PROJECT

Ariel Mundo

2020-12-10

PART 1

Sparse Classification

Comparing penalized methods for High-dimensional Classification

We would like to compare the performance of several modern regression methods including penalized regression methods such as LASSO and Elastic Net regression with 10-fold cross-validation.

Library hdrm was installed following the directions from: <https://github.com/pbreheny/hdrm>

```
library(hdrm)
```

```
## Loading required package: glmnet
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.0-2
```

```
## Loading required package: ncvgreg
```

```
library(glmnet)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages ----- tidyverse 1.3.0 --
```

```
## v tibble 3.0.4    v dplyr 1.0.2
## v tidyr 1.1.2     v stringr 1.4.0
## v readr 1.4.0     v forcats 0.5.0
## v purrr 0.3.4
```

```
## -- Conflicts ----- tidyverse_conflicts() --
```

```
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x tidyr::pack()    masks Matrix::pack()
## x tidyr::unpack() masks Matrix::unpack()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
```

```
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
```

```
##
```

```
## lift
```

```
downloadData(Golub1999)
```

```
attachData(Golub1999)
```

```
n=nrow(X)
```

```
#split in train and test sets
```

```
set.seed(1234)
```

```
train_rows <- sample(1:n, n/2)
```

```
X.train <- X[train_rows, ]
```

```
X.test <- X[-train_rows, ]
```

```
y.train <- y[train_rows]
```

```
y.test <- y[-train_rows]
```

```
#cross validation
```

```
alpha=0.5
```

```
#Using cross validation for best lambda in all models
```

```
fit.elastic_net<-cv.glmnet(X.train,y.train,family="binomial",type.measure = "class",alpha=alpha)
```

```
fit.lasso<-cv.glmnet(X.train,y.train,family="binomial",type.measure = "class",alpha=1)
```

```
fit.ridge<-cv.glmnet(X.train,y.train,family="binomial",type.measure = "class",alpha=0)
```

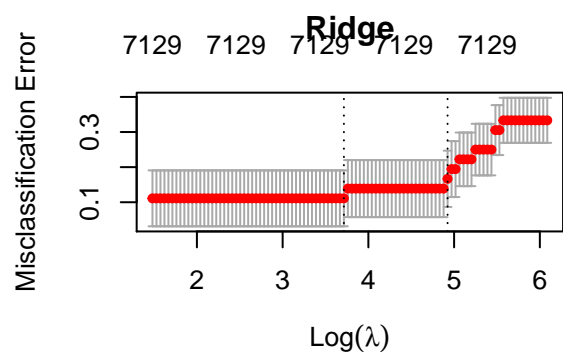
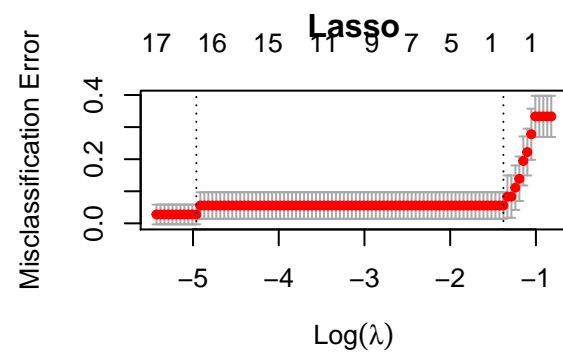
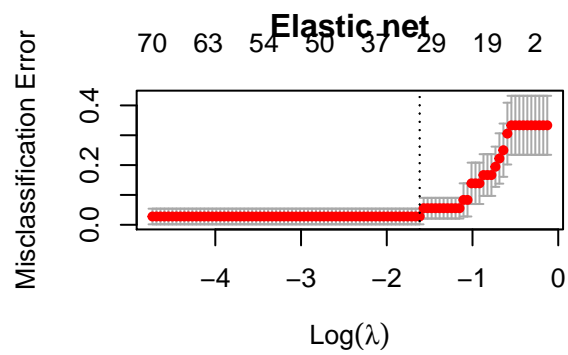
```
par(mfrow=c(2,2))
```

```
plot(fit.elastic_net, main="Elastic net")
```

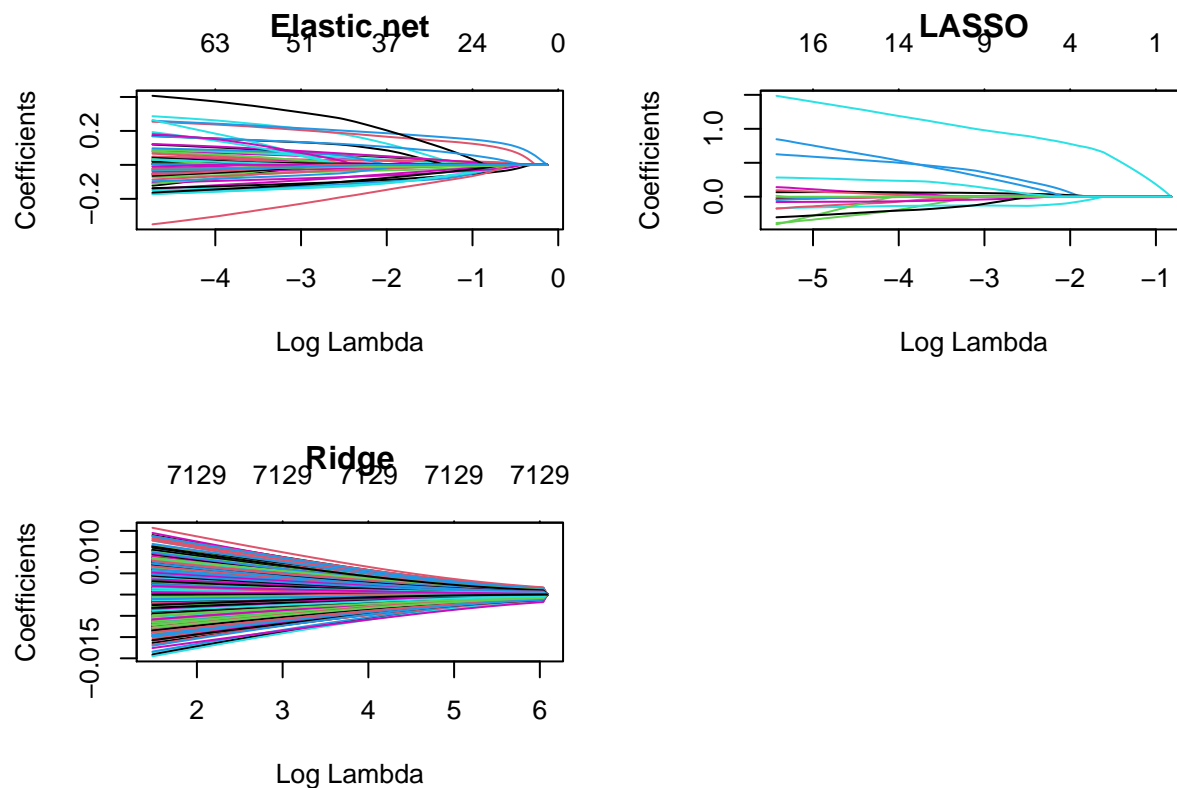
```
plot(fit.lasso, main="Lasso")
```

```
plot(fit.ridge, main="Ridge")
```

```
par(mfrow=c(1,1))
```



```
#Solution paths
par(mfrow=c(2,2))
plot(fit.elastic_net$glmnet.fit,xvar="lambda", main="Elastic net")
plot(fit.lasso$glmnet.fit,xvar="lambda",main="LASSO")
plot(fit.ridge$glmnet.fit,xvar="lambda",main="Ridge")
```



The solution paths are very crowded for the elastic net and ridge regression. Non-zero coefficients will be extracted and sorted in descending order. Because the number of coefficients for Ridge Regression is too high (~7130) only the first 50 will be used for plotting purposes.

```
#making dataframes from the non-zero coefficients
coef.enet<-coef(fit.elastic_net,s="lambda.min")
coef.enet <- data.frame(
  features = coef.enet@Dimnames[[1]][ which(coef.enet != 0 ) ],
  coefs    = coef.enet                [ which(coef.enet != 0 ) ]
)

coef.lasso<-coef(fit.lasso,s="lambda.min")
coef.lasso <- data.frame(
  features = coef.lasso@Dimnames[[1]][ which(coef.lasso != 0 ) ],
  coefs    = coef.lasso                [ which(coef.lasso != 0 ) ]
)

coef.ridge<-coef(fit.ridge,s="lambda.min")
coef.ridge <- data.frame(
  features = coef.ridge@Dimnames[[1]][ which(coef.ridge != 0 ) ],
  coefs    = coef.ridge                [ which(coef.ridge != 0 ) ]
)
```

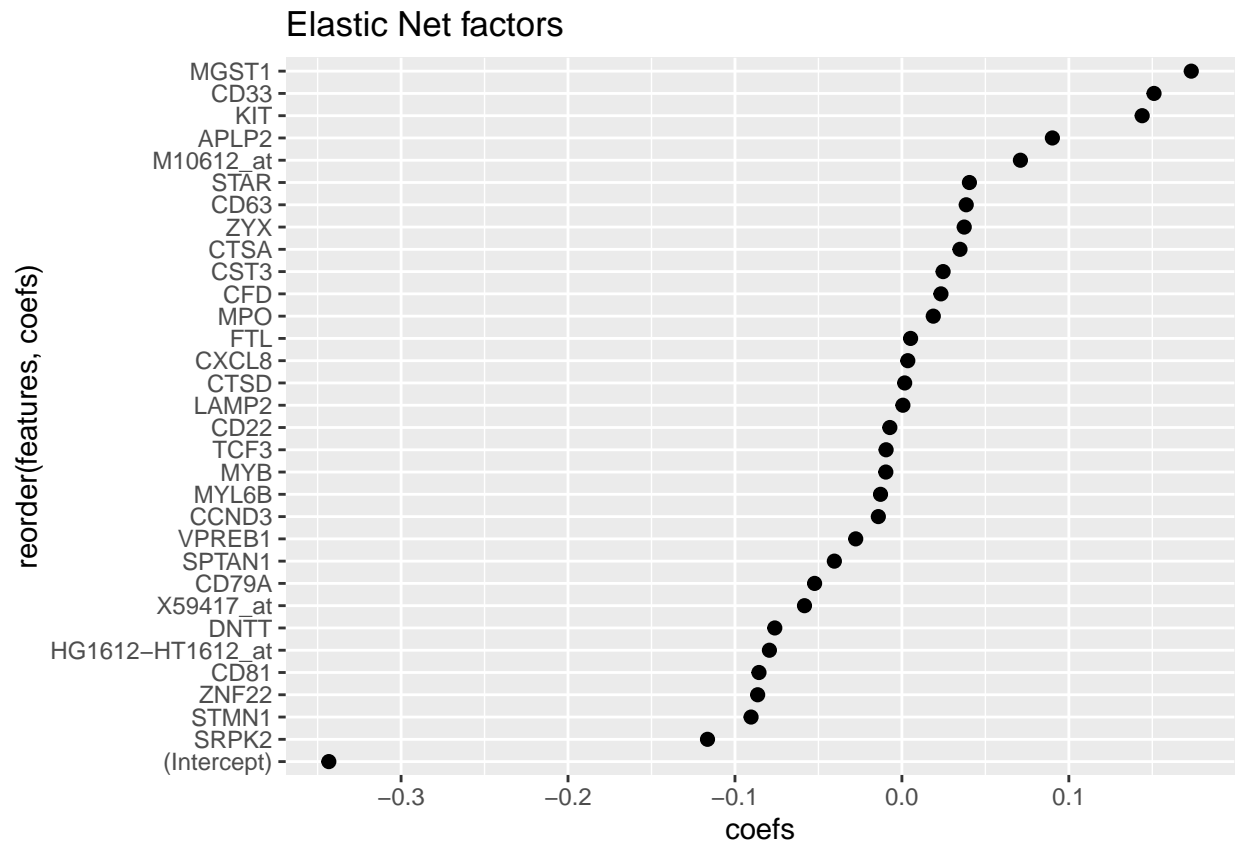
```
#Sorting in decreasing order
```

```
#Because the number of coefficients for Ridge Regression is too high (~7130) only the first 50 will be
```

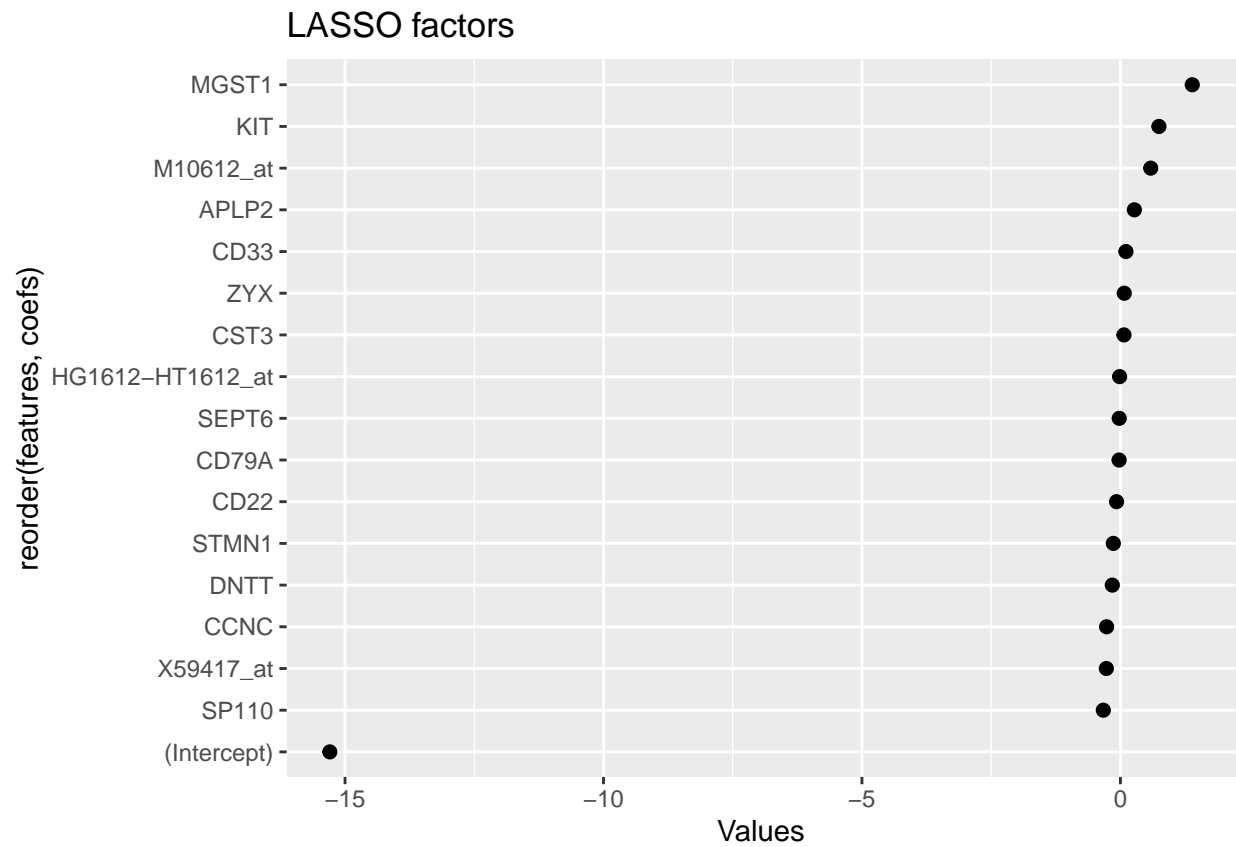
```
coef.ridge_sorted<-coef.ridge %>% arrange(-coefs)
coef.ridge_sorted<-coef.ridge_sorted[1:30,]
```

#making plots with values in descending order

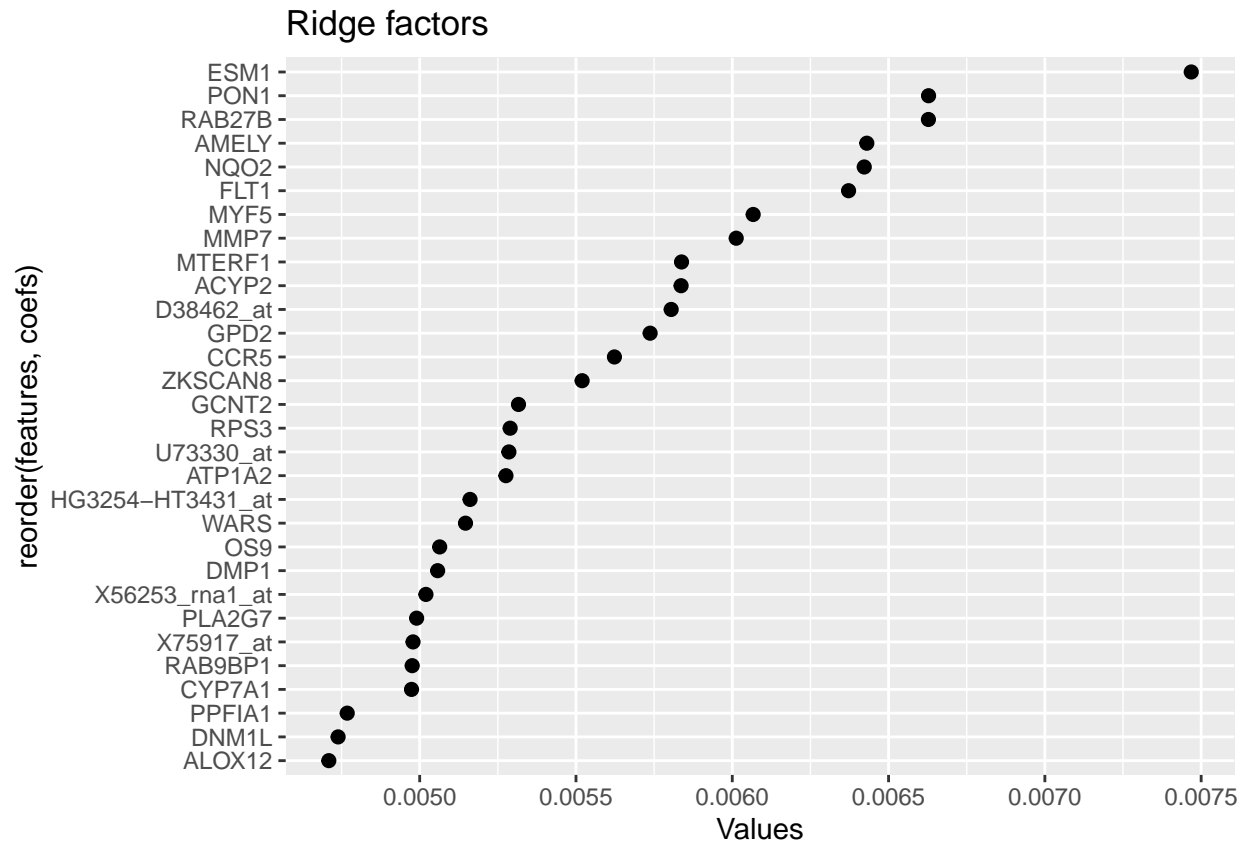
```
ggplot(coef.enet,aes(x=coefs,y=reorder(features,coefs)))+geom_point(size=2)+ggtitle('Elastic Net factors')
```



```
ggplot(coef.lasso,aes(x=coefs,y=reorder(features,coefs)))+geom_point(size=2)+ggtitle('LASSO factors')+1.
```



```
ggplot(coef.ridge_sorted,aes(x=coefs,y=reorder(features,coefs)))+geom_point(size=2)+ggtitle('Ridge factors')
```



```
#coefplot(fit.elastic_net,lambda='lambda.min',sort="magnitude")
#coefplot(fit.lasso,lambda='lambda.min',sort="magnitude")
#coefplot(fit.ridge,lambda='lambda.min',sort="magnitude")

a<-intersect(coef.ridge$features,coef.enet$features) #finding common coefficients between models
a<-intersect(a,coef.lasso$features)
```

It is interesting to note that non-zero factors between Elastic Net and LASSO follow roughly the same descending order, but that is not the case for Ridge. It can be seen that Elastic Net uses 32 significant coefficients, LASSO uses 17 and Ridge uses 7130.

```
#Fitting the models to the training data

mod.enet<-glmnet(X.train,y.train,alpha=0.5,family = "binomial",lambda=fit.elastic_net$lambda.min)
mod.lasso<-glmnet(X.train,y.train,alpha=1,family = "binomial",lambda=fit.lasso$lambda.min)
mod.ridge<-glmnet(X.train,y.train,alpha=0,family = "binomial",lambda=fit.ridge$lambda.min)

#Number of significant (non-zero) coefficients per model for the training set
```

The predictions will be analyzed using the package caret for determining the Confusion Matrix and other statistical information of the predictions.

```

#Predictions on training data
prob.enet<-mod.enet %>% predict(newx=X.test,s=fit.elastic_net$lambda.min,type="response")
prob.lasso<-mod.lasso %>% predict(newx=X.test,s=fit.lasso$lambda.min,type="response")
prob.ridge<-mod.ridge %>% predict(newx=X.test,s=fit.ridge$lambda.min,type="response")

#Because from the contrast matrix level 0 is ALL and level 1 is AML, the probabilities will be re-coded

contrasts(y)

```

```

##      AML
## ALL    0
## AML    1

```

```

pred.class.enet<-ifelse(prob.enet>0.5,"AML","ALL")
pred.class.lasso<-ifelse(prob.lasso>0.5,"AML","ALL")
pred.class.ridge<-ifelse(prob.ridge>0.5,"AML","ALL")

```

```

confusionMatrix(data=as.factor(pred.class.enet),reference=y.test)

```

```

## Confusion Matrix and Statistics
##
##              Reference
## Prediction ALL  AML
##      ALL  22   1
##      AML   1  12
##
##              Accuracy : 0.9444
##              95% CI : (0.8134, 0.9932)
##      No Information Rate : 0.6389
##      P-Value [Acc > NIR] : 2.202e-05
##
##              Kappa : 0.8796
##
##      McNemar's Test P-Value : 1
##
##              Sensitivity : 0.9565
##              Specificity : 0.9231
##              Pos Pred Value : 0.9565
##              Neg Pred Value : 0.9231
##              Prevalence : 0.6389
##              Detection Rate : 0.6111
##      Detection Prevalence : 0.6389
##              Balanced Accuracy : 0.9398
##
##              'Positive' Class : ALL
##

```

```

confusionMatrix(data=as.factor(pred.class.lasso),reference=y.test)

```

```

## Confusion Matrix and Statistics
##

```



```

##           Reference
## Prediction ALL AML
##      ALL  22   1
##      AML   1  12
##
##           Accuracy : 0.9444
##           95% CI : (0.8134, 0.9932)
##      No Information Rate : 0.6389
##      P-Value [Acc > NIR] : 2.202e-05
##
##           Kappa : 0.8796
##
##  McNemar's Test P-Value : 1
##
##           Sensitivity : 0.9565
##           Specificity : 0.9231
##      Pos Pred Value : 0.9565
##      Neg Pred Value : 0.9231
##           Prevalence : 0.6389
##      Detection Rate : 0.6111
##      Detection Prevalence : 0.6389
##      Balanced Accuracy : 0.9398
##
##      'Positive' Class : ALL
##

```

```

confusionMatrix(data=as.factor(pred.class.ridge),reference=y.test)

```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction ALL AML
##      ALL  23   2
##      AML   0  11
##
##           Accuracy : 0.9444
##           95% CI : (0.8134, 0.9932)
##      No Information Rate : 0.6389
##      P-Value [Acc > NIR] : 2.202e-05
##
##           Kappa : 0.8754
##
##  McNemar's Test P-Value : 0.4795
##
##           Sensitivity : 1.0000
##           Specificity : 0.8462
##      Pos Pred Value : 0.9200
##      Neg Pred Value : 1.0000
##           Prevalence : 0.6389
##      Detection Rate : 0.6389
##      Detection Prevalence : 0.6944
##      Balanced Accuracy : 0.9231
##
##      'Positive' Class : ALL
##

```

```
##
```

```
#table(y.test,pred.class.enet)
#table(y.test,pred.class.lasso)
#table(y.test,pred.class.ridge)
#pred.class.enet<-ifelse(prob.enet>0.5,"ALL","AML")
#pred.class.lasso<-ifelse(prob.lasso>0.5,"AML","ALL")
#pred.class.ridge<-ifelse(prob.ridge>0.5,"AML","ALL")

#Misclassification error

1-mean(pred.class.enet==y.test)
```

```
## [1] 0.05555556
```

```
1-mean(pred.class.lasso==y.test)
```

```
## [1] 0.05555556
```

```
1-mean(pred.class.ridge==y.test)
```

```
## [1] 0.05555556
```

The results are similar with either model regarding accuracy (94.4%) and misclassification error (5.5%). It is interesting to note that virtually the same results can be obtained

```
real<-as.data.frame(y.test)
results<-cbind(pred.class.enet,pred.class.lasso,pred.class.ridge,real)
names(results)[1]<-"Elastic_Net"
names(results)[2]<-"LASSO"
names(results)[3]<-"Ridge"
names(results)[4]<-"Real_values"
results$subject<-rownames(results) #adding the correct subject number to each classification
view(results)
```

It can be seen that the Elastic Net and LASSO models misclassify observations 17 and 32. Patient/subject 17 is ALL but the models classify it as AML. Patient 32 is AML but is classified as ALL. The properly classified ALL and AML observations will be analyzed and compared with the misclassified results to determine the cause of the error.

```
#Merging test data
test_dataset<-as.data.frame(X.test)
test_dataset<-cbind(results[,5],results[,4],test_dataset)
names(test_dataset)[1]<-"Subject" #renaming the columns
names(test_dataset)[2]<-"Class"

#Reshaping data
library(reshape)
```

```
##
```

```
## Attaching package: 'reshape'
```

```
## The following object is masked from 'package:dplyr':
##
##   rename
```

```
## The following objects are masked from 'package:tidyr':
##
##   expand, smiths
```

```
## The following object is masked from 'package:Matrix':
##
##   expand
```

```
long<-melt(test_dataset,id=c("Subject","Class"))
names(long)[3]<-"Gene" #renaming the columns
names(long)[4]<-"Val"
```

```
means<-long %>% group_by(Gene,Class) %>% #calculating mean values for each gene per class
  summarise_at(vars(Val),funcs(mean))
```

```
## Warning: 'funcs()' is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with 'tibble::lst()':
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_warnings()' to see where this warning was generated.
```

```
#Getting mean values for the most significant genes per group
sig_means<-means %>% filter(Gene %in% c("MGST1","KIT","M10612_at","APLP2","CD33","ZYX"))
```

```
#extracting values for the missclassified subjects
```

```
vals<-long %>% filter(Subject %in% c("17","32") & Gene %in% c("MGST1","KIT","M10612_at","APLP2","CD33"))
```

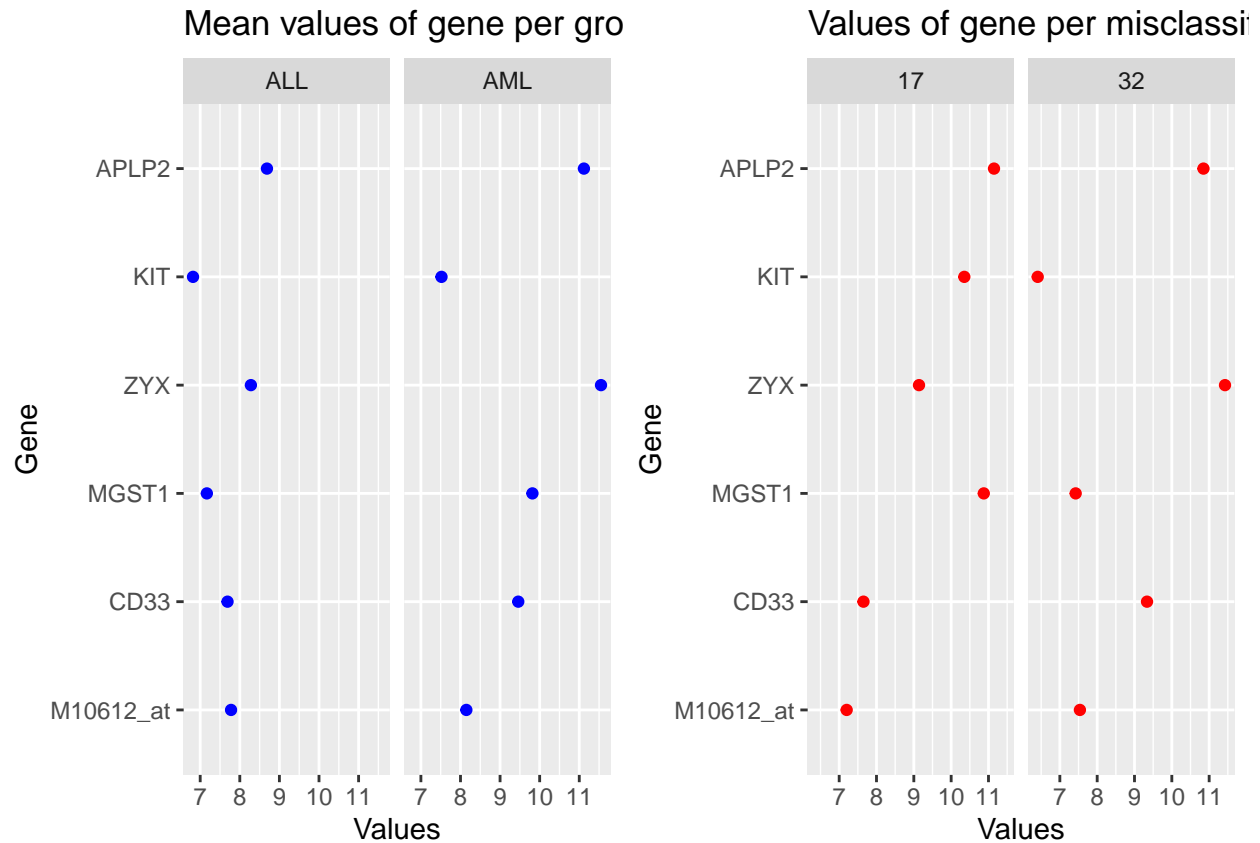
```
g1<-ggplot(sig_means,aes(x=Val,y=Gene))+geom_point(color="Blue")+facet_wrap(~Class)+
  ggtitle("Mean values of gene per group")+labs(x='Values')
```

```
g2<-ggplot(vals,aes(x=Val,y=Gene))+geom_point(color="red")+facet_wrap(~Subject)+
  ggtitle("Values of gene per misclassified subject")+labs(x='Values')
```

```
library(ggpubr)
```

```
#Plotting results together
```

```
ggarrange(g1,g2,ncol=2,nrow=1)
```



Based on this analysis, it appears that the reason why subject 17 is misclassified as AML and subject 32 is misclassified as ALL is because the coefficients associated with the most important genes are closer to the mean values for the other class. It can be seen that the means for AML for the genes tend to be around 8 and up to about 12. The same spread pattern is seen in subject 17, although is classified as ALL. For subject 32 although the data is more sparse it can be seen that the gene values are also below 9, similarly to the ALL group means, although subject 32 is classified as AML.